Optimized Delta Execution for Efficient Mutation Testing

Thiago Vieira
Marcelo d'Amorim

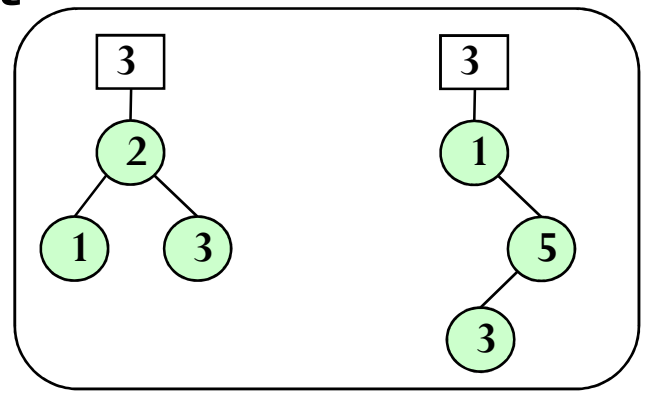
Federal University of Pernambuco, Brazil

SOTA 2011 - Workshop on the State of the Art in Automated Software Engineering Research. Moffet Field-CA, USA

Delta Execution (DE) [ISSTA 2007, TSE 2008]

- Special mode of execution; semantics is based on sets!

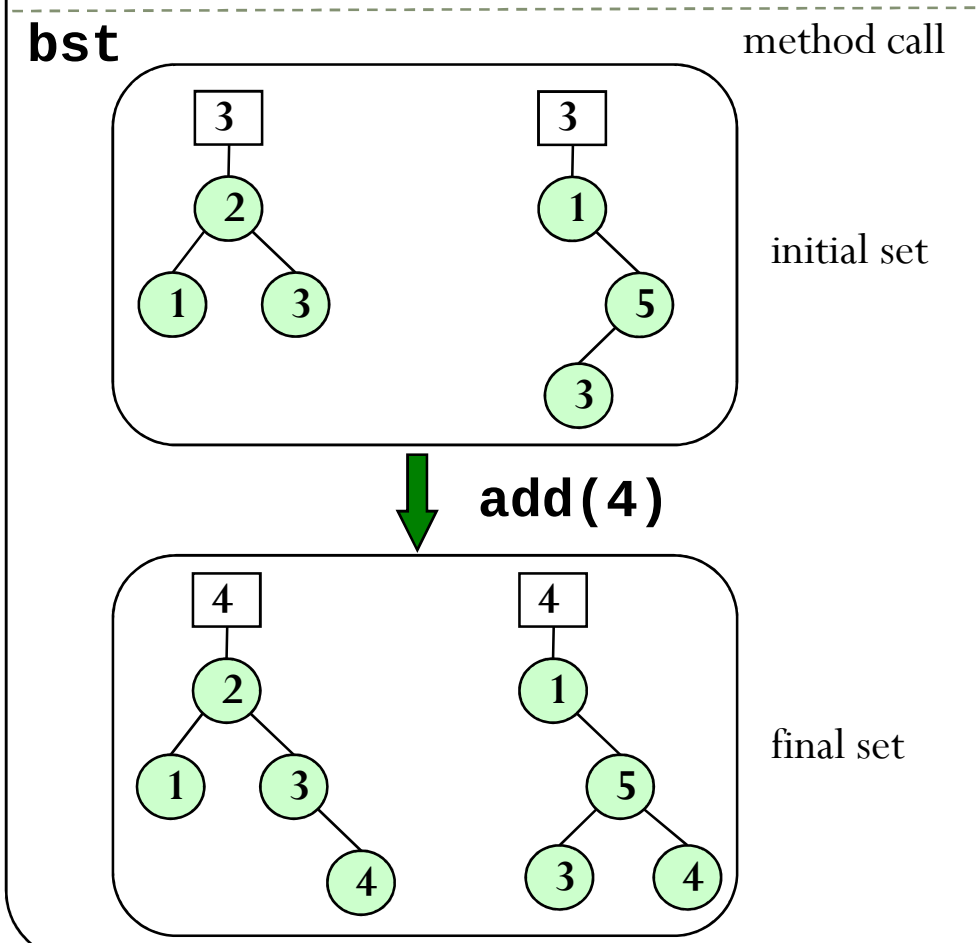
bst



initial set

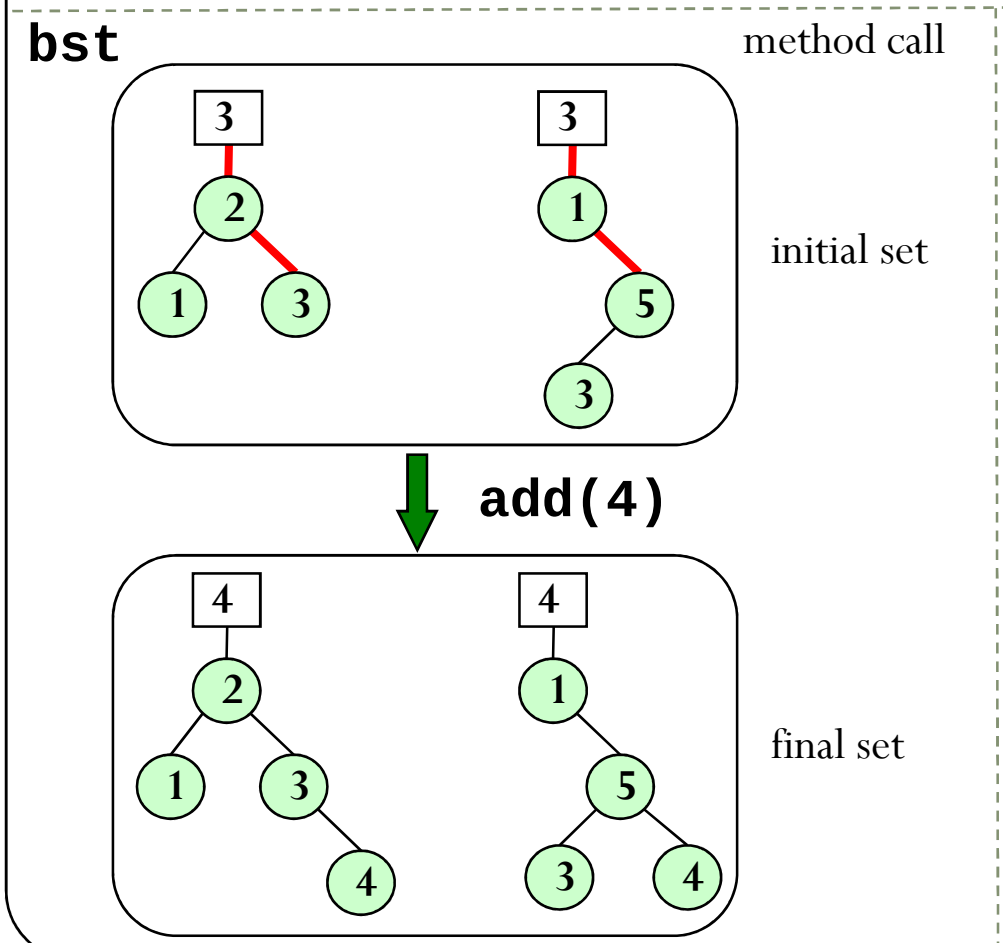
Delta Execution (DE) [ISSTA 2007, TSE 2008]

- Special mode of execution; semantics is based on sets!



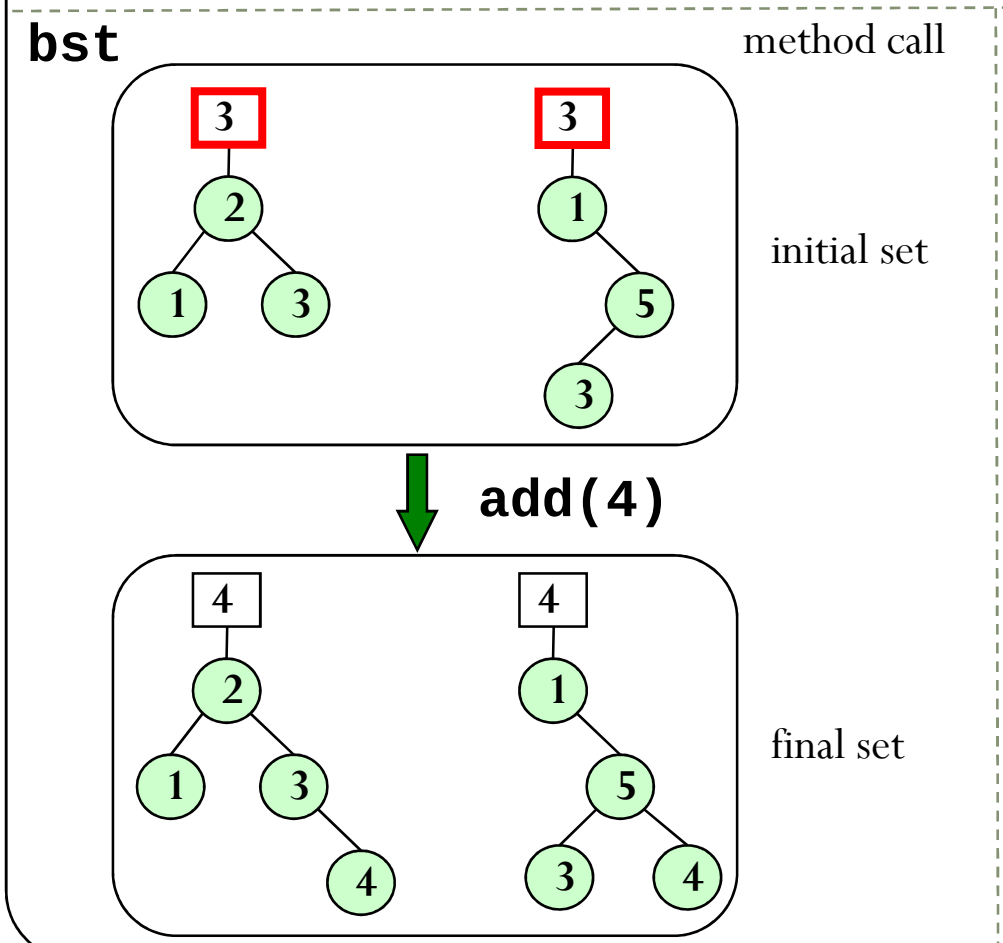
Delta Execution (DE) [ISSTA 2007, TSE 2008]

- Special mode of execution; semantics is based on sets!



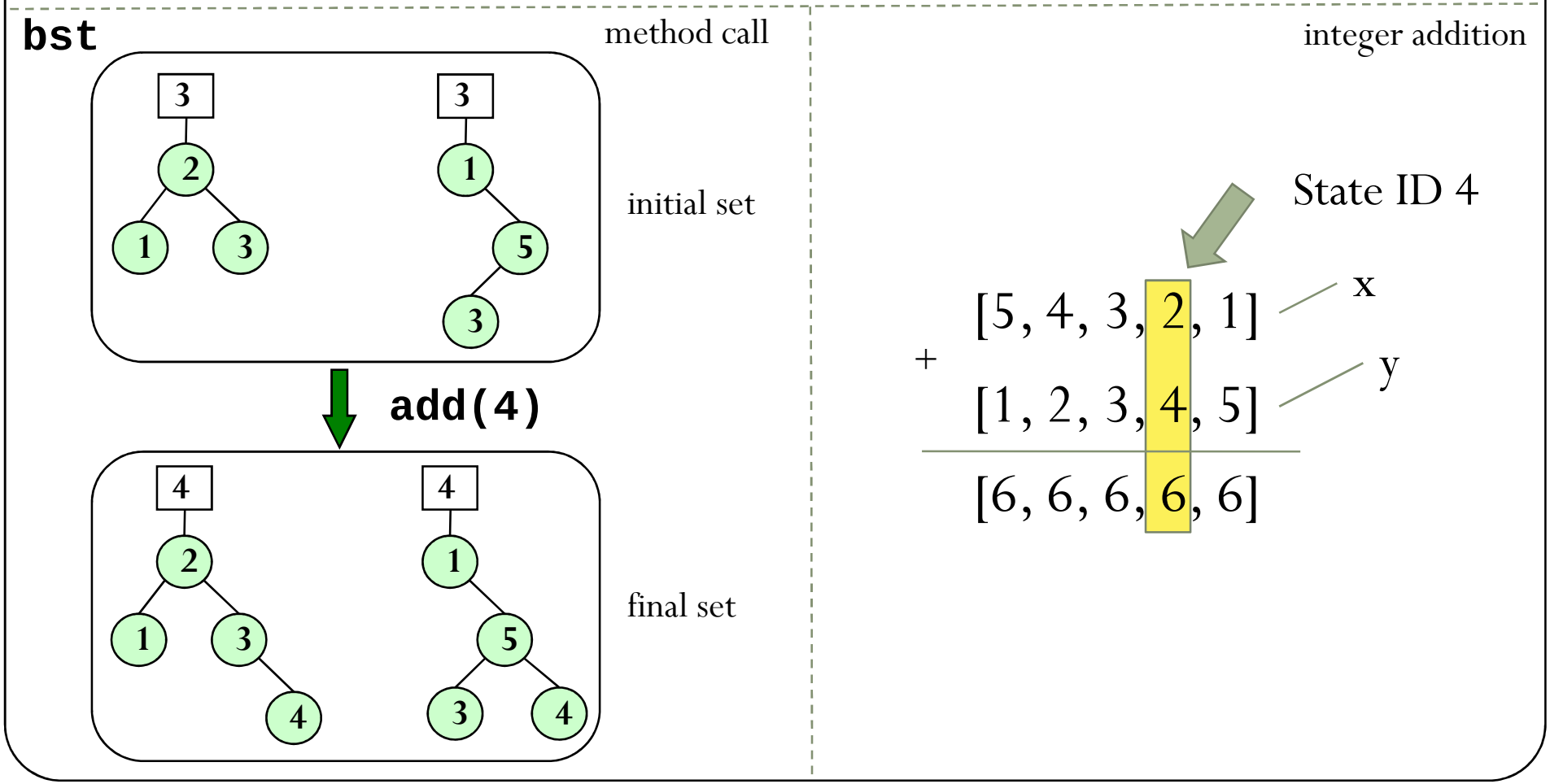
Delta Execution (DE) [ISSTA 2007, TSE 2008]

- Special mode of execution; semantics is based on sets!



Delta Execution (DE) [ISSTA 2007, TSE 2008]

- Special mode of execution; semantics is based on sets!



New Application: Mutation Testing

Problem:
Mutation Analysis is expensive!

- Cost proportional to #mutants x #tests
- Several proposed optimizations [IEEE Software 2010]
  - Javalanche [FSE 2009, ISSTA 2009] runs tests in **parallel** on each **covered** mutant using a **mutant schemata**

Problem: Mutation Analysis is expensive!

- Cost proportional to #mutants x #tests
- Several proposed optimizations [IEEE Software 2010]
  - Javalanche [FSE 2009, ISSTA 2009] runs tests in **parallel** on each **covered** mutant using a **mutant schemata**

Original

```
…else if (tmp.info > info) {
  …
} else …
```

Mutant 5

```
…else if (tmp.info >= info) {
  …
} else …
```

Mutant Schemata

```
…else if (MUT5? (tmp.info >= info) : (tmp.info > info)) {
  …
} else …
```

Problem:
Mutation Analysis is expensive!

- Cost proportional to #mutants x #tests
- Several proposed optimizations [IEEE Software 2010]
  - Javalanche [FSE 2009, ISSTA 2009] runs tests in **parallel** on each **covered** mutant using a **mutant schemata**

Mutation analysis can still take hours to run even for test suites that run in seconds!

Proposal: Analyze mutants with DE

- Run one test against all mutants **simultaneously**!

Regular Execution

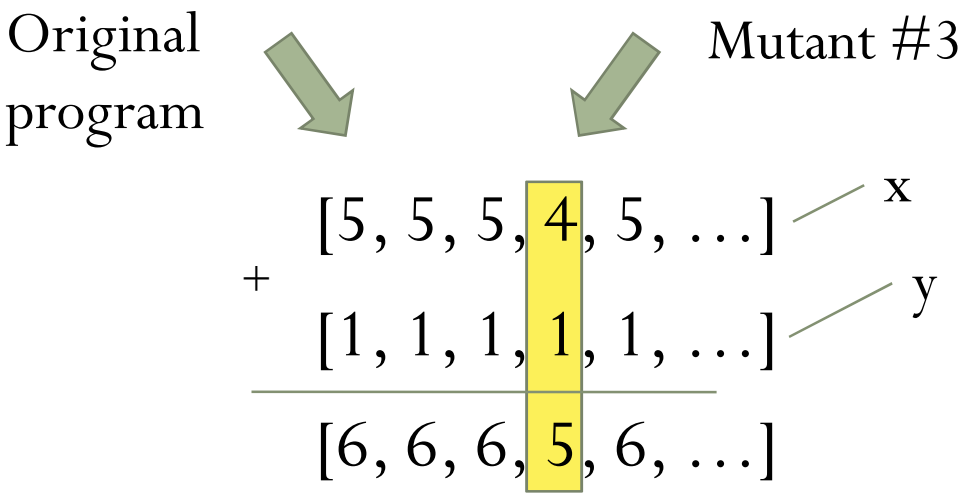
run testOne() on Mutant 1
run testOne() on Mutant 2
…
run testOne() on Mutant M
run testTwo() on Mutant 1
…

Delta Execution

run testOne() on Mutant Schemata
run testTwo() on Mutant Schemata
…

Proposal: Analyze mutants with DE

- The state set contains M + 1 elements
  - M is the number of mutants
  - 1 states identifies the original program

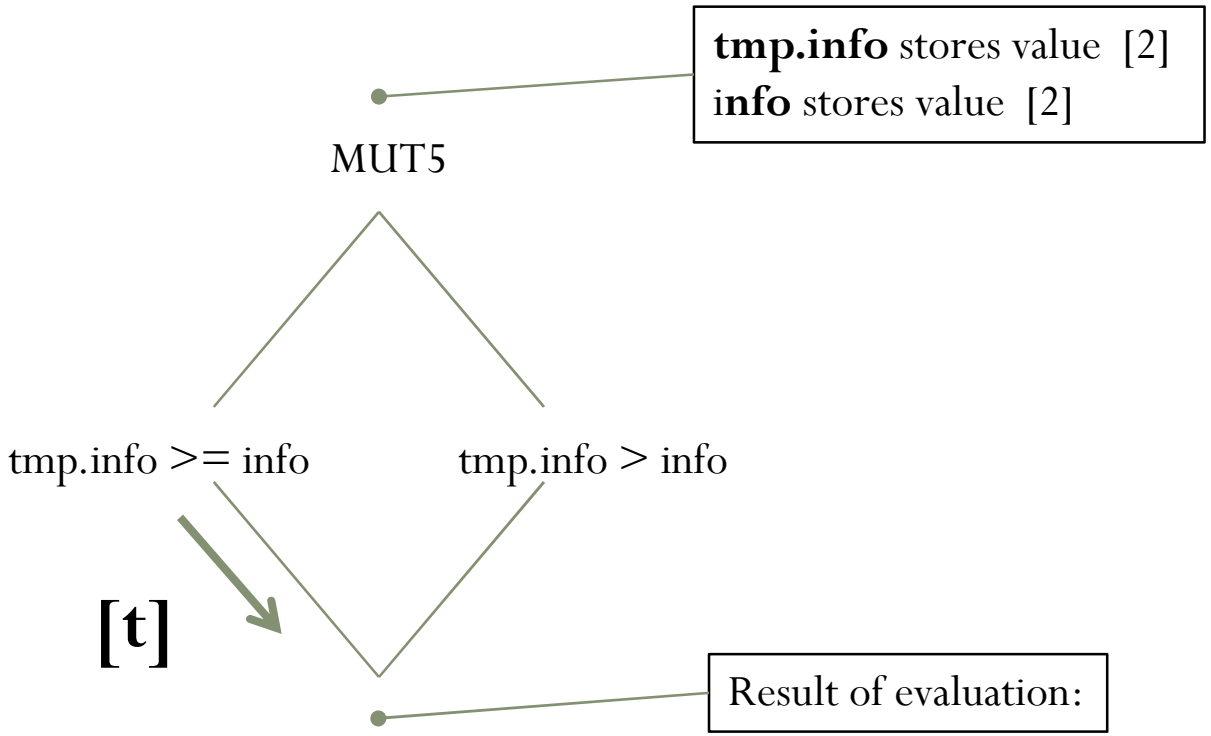


DE at mutation point

Mutant Schemata

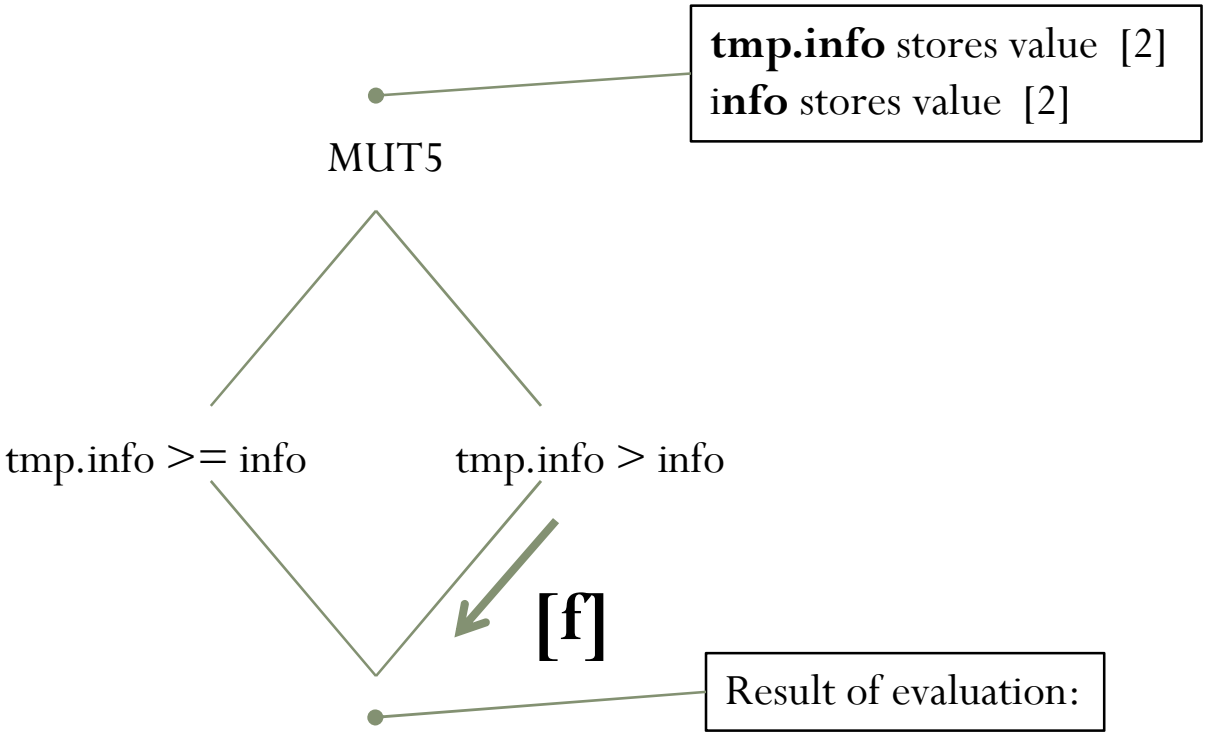
```
…else if (MUT5? (tmp.info >= info) : (tmp.info > info)) {
  …
} else …
```

Example: MUT5? (tmp.info >= info) : (tmp.info > info)



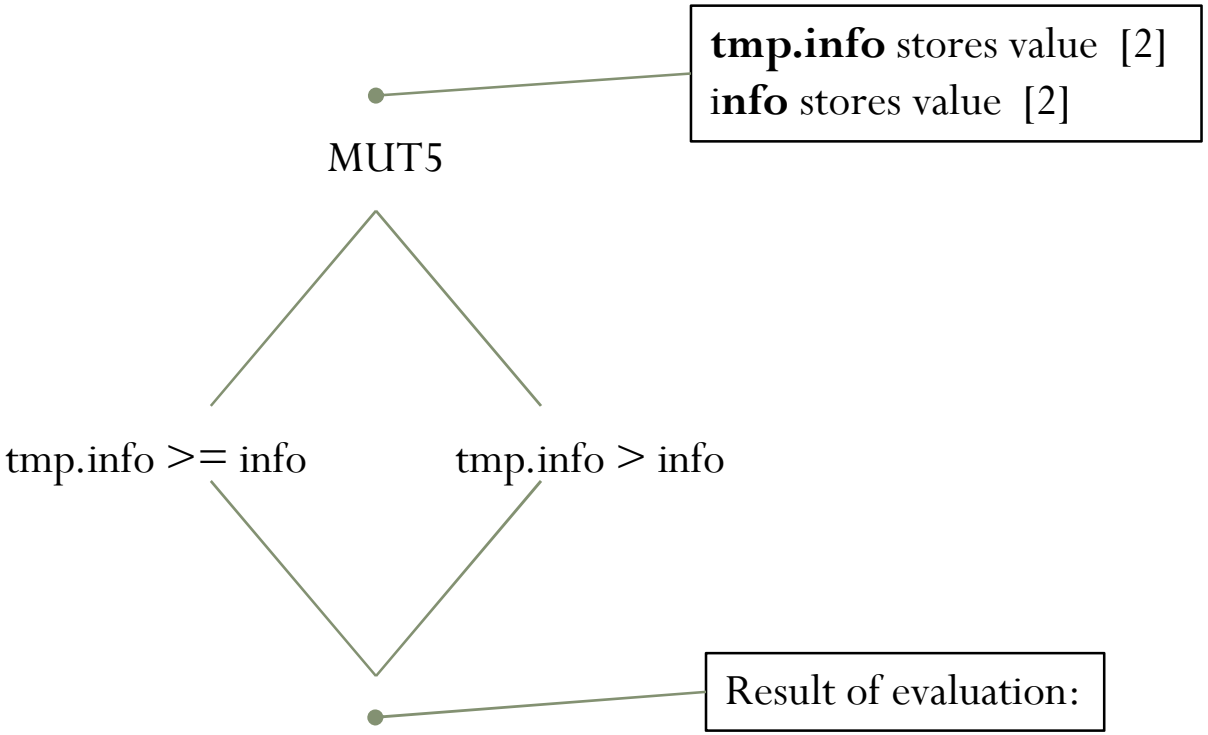
[_, _, _, _, _, t, _, ...]

Example: MUT5? (tmp.info >= info) : (tmp.info > info)



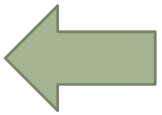

```
[_, _, _, _, _, t, _, ...]
[f, f, f, f, f, _, f, ...]
```

Example: MUT5? (tmp.info >= info) : (tmp.info > info)

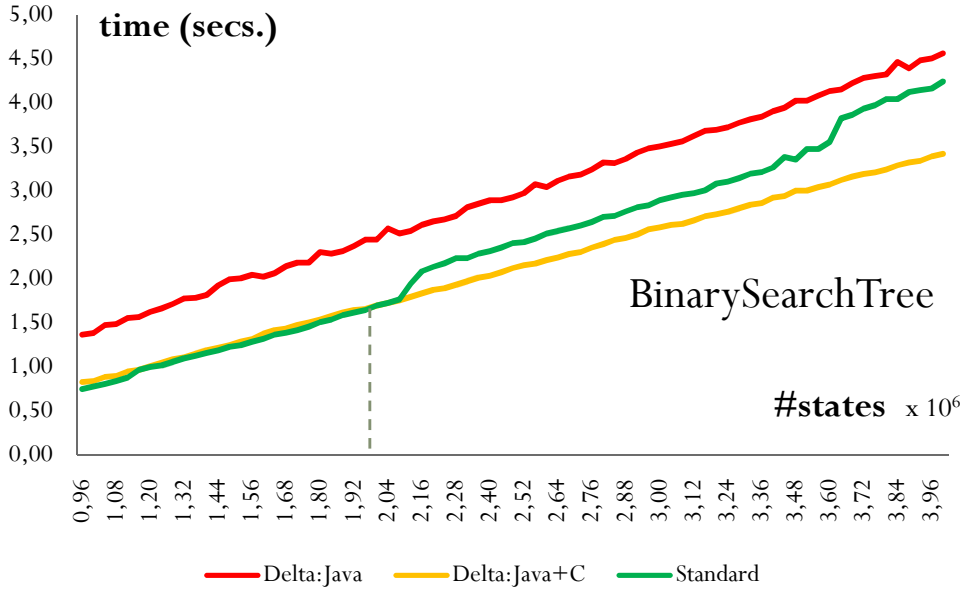


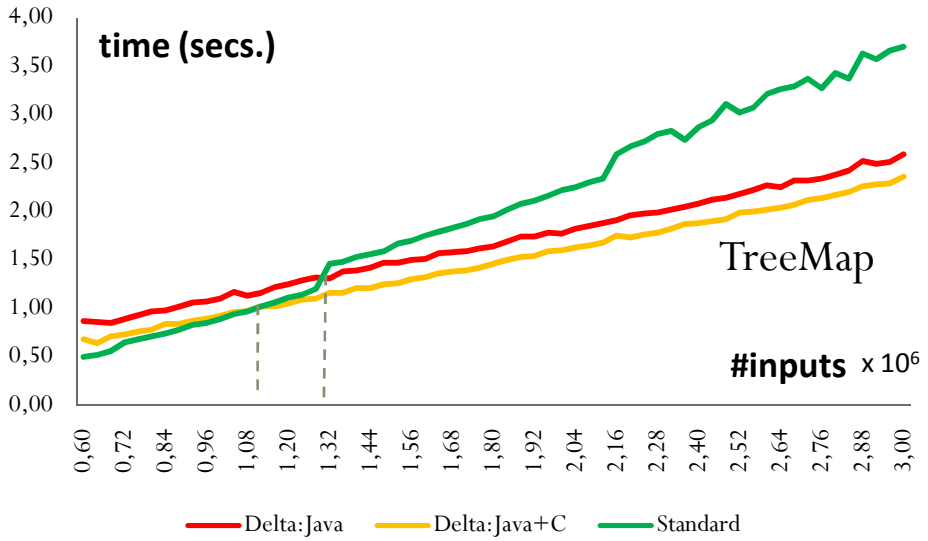

```
[f, f, f, f, f, t, f, ...]
```

DE optimizations

- Mutation specific
  - Ignore states when mutant does not affect state
- General
  - C implementation of Delta operations 
  - Parallelism
    - Parallel exploration of paths
    - Asynchronous evaluation of expressions

Native Implementation of DE





- Setup: execution of method add/put with increasing number of inputs
- Pessimistic scenario
- Ideal scenario:
  - Test execution takes longer
  - Relatively fewer splits

Remarks

- Current Status…
- DE could also help:
  - Assist test generation (e.g., group mutants by change impact)
  - Optimize testing of Software Product Lines

For more information visit:

PAN http://pan.cin.ufpe.br